2001

UML as a foundation to technology

Amy Whiteside

UML as a Foundation for technology

By

Amy Brooke Whiteside

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology

Rochester Institute of Technology

B. Thomas Golisano College
Of
Computing and Information Sciences

November 16, 2001

Rochester Institute of Technology

Abstract

UML AS A FOUNDATION FOR TECHNOLOGY

By Amy Brooke Whiteside

Chairperson of the Supervisory Committee: Professor Timothy Wells
Department of Information Technology

The UML is becoming the recognized standard for object-oriented modeling. The incorporation of this standard into the curriculum of Information Technology academia appears to be a natural progression. How a course can be implemented and in what force is the primary focus for this thesis. In addition to the academic advantages and necessities of the UML as a foundation course.

Rochester Institute of Technology
Department of Information Technology

Master of Science in Information Technology
Thesis Approval Form

Student Name: Amy Brooke Whiteside

Project Title: UML as a Foundation to Technology

Project Committee

Name	Signature	Date
Timothy Wells Committee Member		11/16/01
Jim Leone Committee Member		11/16/01
Brad Rinard Committee Member		11/16/01

UML as a Foundation for Technology

I, Amy B. Whiteside, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my
thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 1-16-2001 Signature of Author: Amy B. Whiteside

TABLE OF CONTENTS

Chapter 1

AN INTRODUCTION

In the past, academic institutions for Information Technology focused on having a programming language as a primary requirement for course work in their programs. As most careers and studies involve some level of software development, for graduates in this field having a solid understanding of programming was a practical initiative.

In fact a concrete awareness of programming became an essential skill in the past five (5) years. Having the concepts of how and why programming was used became almost as essential is being able to actually code. As the technology of programming advanced into object-oriented thinking, there became a need to represent this new facet. Flowcharts were no longer able to handle the detailed systems now capable with OOP and with the wide use of programming and project teams, some formal type of documentation was needed.

Still turning out very skilled students into the industry, universities increased their enrollment in technology related fields and kept on churning. However, the skills needed as the technology boom emerged were beginning to change. The universities had a hard time keeping up with the industry changes that came about. Soon technology and computers began to show their faces in more areas than in the past. Careers that once did not require the need for computer skills advanced into IT-related fields, which required an understanding of what was possible with technology and a general knowledge of how to get it.[i]

Soon emerged the idea of somebody to manage these IT professionals as their role and workload kept increasing. The Chief Information Officer or IT Director soon came to be. Though the industry made the leap to recognize the need for such a position, the precise responsibilities of the position were uncertain. The need for technologically savvy professionals increased and the responsibilities for them evolved as the positions did.

The idea of Information Technologists became a desirable phenomenon. Professionals throughout the industry mended the information of various departments together to operate in a manner useful to the company as a whole. They became responsible for not only the hardware involved to make such things happen but also the software development of how to make it take place. These tasks required technical skills and universities and academic institutions honed in on these skills and began to mass-produce programs and students from them. As the function of the IT professional continued to grow it became evident that their role in the company was vital. With the increase in business critical responsibilities that IT-related employees began to see, the complementary tasks that came along with those roles were not as familiar as their technological task-driven duties prior.

The manifestation of the position that Information Technology played in a company changed the skill set that professionals in the industry needed to be successful. IT professionals began to get involved on planning and strategic positioning of the company based upon the technology and usefulness of the data within the company. Scientific and task-driven programming was no longer enough to survive if the professional had plans of growing with the company. IT professionals soon began discussing projects with customers and gathering requirements for projects that would then turn into business applications that drove the company.

The skill set focused on by many universities provided a wealth of information for students on technical aspects of the career. However, with the changing role of the IT professional came new responsibilities that went outside the scope of task-driven, scientific technology.

I propose that one of the changes that many universities failed to incorporate into the curriculum for Information Technology is the standard modeling language in the software development industry – the Unified Modeling Language.

Chapter 2

A BACKGROUND

In order to examine the advantages of implementing a foundation course such as the Unified Modeling Language (UML) into the academic preparation for programmers and systems analysts, it is important to first look at the UML itself. This introduction chapter will give a brief overview of the UML – its origin, history and purpose.

The Unified Modeling Language (UML) was adopted as a standard by the Object Modeling Group (OMG) in 1997.[ii] The OMG was founded and created by eleven (11) large industry-leading companies in 1989. The purpose of the OMG was to produce industry specifications for object software in order to create a component-based industry. (See Appendix A. The OMG Mission)[iii] The UML is "a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system."[iv]

As stated in Chapter 1, systems and projects dependent on the IT professional have changed over time. Along with these system changes comes the need for changes in development. System development no longer refers to one system being implemented without regard to the rest of the company. Integration between business systems has become the de facto for information systems development. Systems are not developed to be independent processes or stand alone applications in the changing industry environment.

Due to the multiple systems involved, integration between these systems can cause problems. Another issue to think about is ad-hoc development. Ad-hoc development tends to be done with a particular purpose in mind, which doesn't allow for flexibility or for growth. These developments are often times temporary and lead to further problems down the road. Due to the lack of pre-defined architecture involved in planning, firms have become accustomed to bottom-up approaches. As with the situation ad-hoc development creates, the same holds true with bottom-up approaches. This type of approach does not take into consideration change and growth and causes more problems to occur.[v]

It is important to remember that the UML is a modeling language, not a method. As the stated in the definition, a modeling language is a graphical representation used to express the specifications of a system. Whereas a method incorporates a process along with this graphical depiction to describe what steps to take in creating the specification. By keeping the UML strictly a modeling language, any process can be applied which increases its flexibility and autonomy. Although the language was not developed to work explicitly with a specific process, it is recommended and optimal if a process is applied to the use of the UML[vi]

Modeling is important in many avenues of industry including software development. Modeling aids in the organization and plan of software. In many disciplines outside of Information Technology, modeling is a very common practice. Take for example engineering. In the engineering field models are developed for multitudes of purpose. They are used to show clients their end result, an aid for visualizing the structure of a building and even to test mathematical theories for safety sake. The software industry has been slow to accept the idea of modeling into practice, however it is taking steps towards the progress and industry maturity that modeling offers.

Booch, Jacobson and Rambaugh describe a model as a simplification of reality -- a simplification, which aids in a better understanding of the system being developed. There are four goals of modeling:

1. Help in visualizing a system as it is or as its wanted to be
2. Allow for specification of the structure or behavior of a system
3. Serve as a template that guides in constructing a system
4. Document the decisions made[vii]

Having a modeling language that exercises a standard vocabulary and set of rules for diagramming that focuses on the conceptual and physical representations of a system serves as software developers blueprints. [viii] The vocabulary and rules in the UML are designed to help standardize large complex projects in a manner that allows for ease in creation and reading of the diagrams.

Developers often begin coding a project as they are thinking it over in their minds. And the time between thinking and coding is generally minimal. The planning and documentation phases of development are generally left to later phases when their absence becomes detrimental to the projects success. Many times the reason they play such a vital role in the developments success is it is discovered that something was ill communicated or interpreted differently by team members. The project faults then become more evident than it successes and its time to go back to the drawing board – or most likely sit down to it for the first time.

The UML is both textual and graphical and therefore compliments the different specifications it offers within it to serve as a very comprehensive modeling tool. With proper modeling, ambiguous interpretation is avoided.

Chapter 3

A SET OF SURVIVAL SKILLS

By offering the UML as a foundation course, students not only gain a skill in advanced programming, they become practiced in a skill set that allows them to reach beyond the code they are writing. Writing code and developing software is one proficiency, however, being able to further develop that trade by making it more flexible for modifications or upgrades or by developing in a fashion that allows the system or components within the system to be reused is a true talent. Increasing the capacity of a programmer to see beyond the current project and beyond their cubicle is a survival skill set that every academic institution should aim to achieve.

In an in industry where working in teams is no longer avoidable, engineers and programmers alike must learn how to work together with various cross-departmental employees to get a project completed. Easily said, but not easily done. Engineering and technical fields, such as computer science and information technology, are rarely formally taught the fundamentals of the socio-cultural aspects of their professions. Gray and Lawson argue that there are two sides of a project, the technical and the socio-cultural. The socio-cultural side of a project deals in part with teams and the "art" of leading and management.[ix]

As Stuart Walesh states in the Journal of Management in Engineering, undergraduate engineering programs frequently return graduates that are excellent technical engineers, but lack the "social science" background to produce engineering managers.[x] "[T]he portrayal of the engineer as a doer rather than a

director or decider, or as a producer rather than a manager or leader" is the problem, not the technical courses being taught.[xi] This also holds true for being part of a team where management creates a creative and initiative-based environment to work in. It is important to realize that although not appointed as a project manager, many teams have technical leads that require similar traits.

One of the most distinguishing skills that UML provides to its users is *communication.* When exploring the UML and its communication benefits, there are many analogies that are used to drive the point home. Anywhere from the advantages of blue prints for engineers[xii] to musical scores for orchestras[xiii], the notion that UML serves as a communication tool between members of a development team and a tool for communicating with customers is clear. When working on a large software project in a team situation, it is essential that everyone on the team have a clear communication strategy in order to create a system where each individual's component comes together as a functioning and efficient solution for the customer. And even more difficult, to create that system in a manner that is usable and accurate in reflecting the customers needs.

In the introduction chapter of "UML Distilled", Fowler validates that when the aim is communication and discussion in the analysis and design phases of software development, it is not the method used to reach the final design that must be understood, rather the design itself that is understood.[xiv] UML lends itself to this communication throughout the life cycle of software development and systems analysis and design due to the multiple views it can provide.

For example, the Use Case View can supply a view of the systems behavior from a user prospective. This type of view serves as communication for what the system will do. The Design View on the other hand delivers the vocabulary and functionality of the system from a static perspective. Performance, scalability and throughput can be seen from the Process View while the system assembly and

configuration management is shown in the Physical or Implementation View. The actual delivery and installation of the system is shown through the Deployment View.[xv]

Any proficient programmer can create a program for himself or herself that works perfectly for their needs. When developing a project for themselves, analysis, design, development, and satisfaction is all determined by one person. However, enterprise development requires the coordination of multiple people skilled in varying areas of the project. Generally in the development of such large systems the condition of satisfaction is not determined by the development team alone, rather it is specified by a customer who is not part of development -- either external or internal to the company.[xvi] Miscommunication to either the customer or members in the development team for what the system will do or the specifics of how it is to come together is certain failure for a project. Communication can be the unwavering solution to a highly complex and involved project. The language to express such communication is made effortless with UML.

With the increase in system functionality and the general decrease in the time-to-market that most companies are faced with, system developers are required to use some type of rapid development techniques to meet the deadlines set before them. As discussed previously, communication can be a key factor to attributing success to a project. However, when faced with a changing environment where team members are often added and taken away from projects, there becomes a need to communicate the project needs and requirements over and over again.

Documentation is crucial to a project throughout the development and maintenance phases. Often times on large projects, team members are pulled from projects and new members are added. Documentation can be the key to ensuring that the projects requirements, scope and goals are communicated properly and with consistency to each team member. Documentation is not only useful to team

members. It also helps to appease upper management and customers when issues such as requirements changes and requirements satisfaction issues arise.

Similar to the diagrams and views noted above with communication, documentation allows for both overall and detailed descriptions of the project. Fowler discusses that the detailed feature intense sections of the system (mainly those used by the developers) can be left up to the documentation of the code. Documentation generation has become a popular tool for many programmers. A few extra keystrokes during code development and with the click of a button a detailed document of the project code is presented. In "UML Distilled", it is recommended that UML diagrams be presented as supporting documentation to such code documentation. A supplemental document may accompany to highlight important ideas within the project. Such documentation should be brief and to the point. Most of the information needed to pick up on the project from a developer's perspective should be included within the UML diagrams and supported by the code itself.[xvii]

Communication and documentation benefits of the UML go hand in hand. The UML diagrams that are essential for documentation in a project are also extremely useful in communicating the project and its uses. The diagrams are not only useful for the development team itself, but practical for communicating complex systems to end-users.

In the industry today there is always a constant question of what technology to use on what platform. The questions are certainly valid ones and need to be addressed when considering the development and performance of large projects. UML offers an initiative to consumers that is unbiased to the questions of what technology and what platform projects are developed in. *Flexibility* is a skill that can save not only money but also time. Ideally requirements would remain the same and system analyst wouldn't miss a beat when determining them. However,

it is more likely that something will get left out or needs changed by the time a project is complete rather than not.

Project teams must balance the consequences of adding features for the sake of schedule, cost, and resources throughout the project life cycle. Not only internal concerns add to the consequences, but also keeping customers happy as well as gaining competitive advantage by features that could be added.[xviii] Weighing the scale between keeping a customer happy and keeping the schedule and budget on target are difficult tasks to manage.

Flexibility with the UML can be seen in various segments of the development process; flexibility in the sense of technology, development of the technology, and the process to which it is completed. With many of the visual UML modeling tools available, namely Rational Rose, changes in a project can be done with little effort. Not to say that changes in a complex system can be easy by any means, however, the tools available for UML allow for changes in one area to be documented and updated through the other areas of your project. It allows for a visual and documented change process along with identify all of the areas that one change will affect.

In today's shortened development lifecycle and increase in system requirements, *reuse* plays a significant part in development. Ian Sommerville states that "[the] demand for lower software production and maintenance costs along with increased quality can only be met by widespread and systematic software reuse."[xix] Software reuse allows students to begin their studies with practices that are used throughout the industry to save time and money. Many times in academic situations, projects are well suited for demonstrating knowledge of a programming language or concept but they are not suited for industry use. Many times the reason for the ill suited software is because no part of the program can be reused.

The UML foundation course would show students from the onset that it is important to keep efficiency, both time and money, in mind when developing commercial software. By keeping these standards in the academic setting, students are gaining tremendous skill at continuing it as accepted practice once out of the program.

Chapter 4

A COURSE

A foundation course for IT programs concentrating in the UML will provide a set of survival skills that are not only necessary for the industry but will also allow subsequent IT classes to communicate on a much higher level. The same set of survival skills detailed in Chapter 3 relates to academic studies as well as they do to the industry. Therefore, having a course that emphasizes communication, documentation, flexibility and reuse allows higher level academic classes to have more clear and in depth study relating to the IT field.

By requiring students to have a UML background, IT programs gain tremendous strength. The advantages lie in both the entry-level classes as well as the upper level course offerings. A foundation course such as the UML offers students an introduction to object oriented thinking without the intricacies of hard-core programming. Beginning students with a modeling language first allows both students and professors to think and communicate with object-oriented tools before beginning to code. This understanding trains students to plan out an approach to programming or designing a system prior to detail-leveled development.

The true advantages of the foundation course are seen throughout the advanced IT courses. By starting with a foundation that creates a development standard for language and diagramming, advanced courses are able to go into more complex programming scenarios and communicate ideas with more clarity than could be done without a standard language.

The UML has become an accepted standard in the industry and lends itself nicely to the academic arena as well. Although many traditional students entering into IT academic programs have some programming experience from their high school education, introducing the UML as a foundation course would assume no previous programming experience.

A recommended prerequisite for enrollment into the UML course would be Introduction to Information Technology or a like offering. Although the UML course would be taught from the ground up and with the idea that students will not have any formal training in programming, the introduction to IT related topics would be helpful in setting the purpose for the course.

As a foundation course, the UML course would be required for all advanced IT courses. By requiring the course for advanced study, professors would be guaranteed a certain level of understanding of the UML. Professors and students would be able to communicate on the same level without taking the time in advanced courses to establish standard terminology or documentation when teaching advanced concepts or detailing projects.

The UML course would begin by discussing object-oriented concepts to familiarize students with the idea of objects, relationships, attributes and methods. UML In A Nutshell by Sinan Si Alhir gives a comprehensive overview in Chapter 3 Object Orientation (pgs 39 – 68) as a guide to follow preparing students with the appropriate base concepts for moving on to the UML.[xx] Examples and descriptions at this point in the course would have no code relation or programming specifics. Supporting cases for the concepts should be taken out of real world abstractions to ensure all students have a clear understanding of the terminology and concepts behind object-oriented thinking. After reviewing object-orientation, the course would move to an overview introduction of the Unified Modeling Language.

An introduction to the UML would begin with defining its purpose and understanding the need for modeling. Some basic principles of modeling and why modeling is important should also be added to provide students with a rationale and appropriateness of the course. The Unified Modeling Language User Guide by Booch, Jacobson, and Rumbaugh gives an overview in Chapter One Why We Model for explaining modeling while a combination of Chapter Two Introducing the UML from the User Guide along with Chapter One Introduction and Chapter Two The Big Picture from UML In A Nutshell for the overall UML introduction.[xxixxii]

A brief description of development processes for software is also needed. Noting that UML is not a method itself, it is recommended that the UML be applied to some type of process. The recommendation is that different development processes be overviewed and then one chosen to focus on for the remainder of the course when processes are needed to further explain the UML in practical scenarios.

Once the background for the course is established, diagramming and actual use of the UML can begin. As far as presentation and instruction for the UML, it is recommended to first establish terminology and concepts relating to the diagram being taught. By establishing terms and concepts prior to the actual modeling techniques students are able to understand what each diagram is used for and when it is necessary to use each diagram. The Unified Modeling Language User Guide uses the approach of introducing terms, concepts, and then modeling techniques throughout the introduction of each of the UML diagrams.

Once the basic diagrams for Use Cases and Class Diagrams are established, the introduction of a UML tool is essential. The tool not only drives home the idea of how UML diagrams can directly impact the cohesion of team development, it also allows students to leave the course with a useful skill that will assist them in

their continued academic study and also a skill that is very marketable in the industry. The recommended tool for use is Rational Rose. Rational Software (www.rational.com) has an academic program, Software Engineering for Education Development (SEED) that is free to academic universities. "The Rational SEED Program provides colleges and universities software engineering tools and courseware free of charge to enable knowledge transfer to the academic community. Rational Software supports the adoption of Best Practices in educating students with state-of-the-art software technologies for hands-on use by students resulting in good software engineering disciplines."[xxiii]

Rational Rose labs should be presented throughout the course to reinforce the diagrams presented during lecture. Team projects should also utilize the Rational Rose software and incorporate a term long project. To aid the student with the Rational Rose software and creating the UML within the tool, the text Visual Modeling with Rational Rose 2000 and UML[xxiv] is recommended. The suggested retail price for the text is $39.95. This text also provides supplemental information for the purpose of modeling and additional descriptions of the diagrams themselves relating to the tool.

The text recommended for the lab goes through the steps for creating each of the diagrams in the course. A sample system is also described in the beginning of the text and each Rational Rose project is in reference to the initial system. A brief background, problem and problem statement is provided to the student prior to any of the exercises. It is recommended that this text be used for introducing the student to modeling via a tool such as Rational Rose.

To coordinate the lab exercises for the students, it is recommended that the professor create a sample system to build the lab exercises around. A system recommendation would be a hospital information system.

Due to the differences between the structures in academic institutions – quarters versus semesters – percentages have been used to illustrate the time of concentration for the areas that the UML foundation course would offer.

Time	Course Concentration
5%	Object-Oriented Concepts
2.5%	Introduction to the Unified Modeling Language
2.5%	Development Processes
2.5%	Mechanisms and Structure
7.5%	Use Cases
5%	Basic Class Diagrams
5%	Interaction Diagrams
2.5%	Team/Project Introductions
7.5%	Advanced Class Diagrams
5%	Packages and Collaborations
5%	State and Activity Diagrams
10%	Physical Diagrams
40%	Rational Rose Lab Exercises

Recommended textbooks for the UML foundation course would be split between a UML diagramming text and a UML Rational Rose text. For the UML diagramming text there are two suitable choices. The Unified Modeling Language User Guide by Booch, Jacobson, and Rumbaugh and UML In A Nutshell by Sinan Si Alhir are both appropriate selections. The user guide written by the three original designers of the UML is a comprehensive text with a suggested retail value of $48.95. The Nutshell series text is written as a quick reference guide for the UML and has a suggested retail price of $24.95.

Though both texts are appropriate for the areas of study in the course, the UML In A Nutshell is recommended as the student text for the foundation course. The Unified Modeling Language User Guide would be well suited as a supplemental text for both the professor and the student.

Mastery and success in the course would be measured with both projects and exams. Individual and group projects would comprise a large portion of the student's grade, as practical demonstration of the UML is a main goal of the course. In addition to the projects and exams, three quizzes would also be given. Below is the recommended grading criteria for the UML foundation course.

Grade Percentage	Type of Measurement
20%	Exam 1
20%	Final Exam
25%	Individual Projects
20%	Team Project
15%	3 Quizzes

Please refer to the below appendices for course samples:

Appendix B. Course Sample Terminology

Appendix. C. Course Sample Diagrams

Appendix D. Course Sample Slides

Appendix E. Course Sample Rational Rose Labs

Appendix F. Course Sample Quiz Questions

Appendix G. Course Sample Homework Assignment

Appendix H. Course Sample Class Exercise

Chapter 5

A CONCLUSION

Entering into this thesis, I was convinced that a programming language should be paired with the teaching of the UML. Throughout my study with a testing course at Frostburg State University, similar to that proposed in this work, I noticed that the students were preoccupied many times with the code and did not concentrate on the main objectives of the course. After trying different approaches to the concepts, I decided that a tool for the UML would help to enforce the concepts I was trying to deliver. In addition to the tool being helpful in diagramming, it also took the diagrams and converted it into a specified code.

By removing the programming component of the course and adding a strong, industry accepted tool, I believe that students become more focused on the UML and the necessity of accuracy and thought put into the diagramming in order to develop code that is representative of the solution the students were trying to propose. Instead of concerning students with learning syntax in this course, the UML foundation course should focus on the concepts that will enable students to pick up any programming language by simply learning individual language rules. As an entry-level course in an IT academic program, this skill is extremely powerful for students and professors alike.

Professors will be able to notice the advantages of the course and skills it provides in starter programming courses when students already understand the concepts of object-oriented thinking and have also seen actual code created by the diagramming of such concepts with Rational Rose. As students progress

through the program, the skills introduced in the foundation course and nurtured through the subsequent courses in the program become a true asset. By issuing students with a survival skill set outlined in Chapter 3, professors will be able to communicate with students at a level that is understandable yet also comprehensive enough to elaborate details of system problems or opportunities that need to be solved.

The advantages given to the professors in a program that offers UML as a foundation course to IT is directed related to the advantages that the student receives. By exercising the practices in the academic arena, students are acclimated to the methods that will be used in industry as well.

APPENDIX A. THE OMG MISSION

The OMG was formed to create a component-based software marketplace by hastening the introduction of standardized object software. The organization's charter includes the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development. Conformance to these specifications will make it possible to develop a heterogeneous computing environment across all major hardware platforms and operating systems. Implementations of OMG specifications can be found on many operating systems across the world today. OMG's series of specifications detail the necessary standard interfaces for Distributed Object Computing. Its widely popular Internet protocol IIOP (Internet Inter-ORB Protocol) is being used as the infrastructure for technology companies like Netscape, Oracle, Sun, IBM and hundreds of others. These specifications are used worldwide to develop and deploy distributed applications for vertical markets, including Manufacturing, Finance, Telecoms, Electronic Commerce, Real-time systems and Health Care.

OMG defines object management as software development that models the real world through representation of "objects." These objects are the encapsulation of the attributes, relationships and methods of software identifiable program components. A key benefit of an object-oriented system is its ability to expand in functionality by extending existing components and adding new objects to the system. Object management results in faster application development, easier maintenance, enormous scalability and reusable software.

The Object Modeling Group
Mission

http://www.omg.org/news/about/index.htm

APPENDIX B. COURSE SAMPLE TERMINOLOGY

METHODOLOGY - An organized plan that breaks the process (of development) into a series of tasks.

OBJECT-ORIENTED PROGRAMMING - Evolved as a better way to isolate logically related portions of an application...easier to develop, debug, and maintain complex applications.

OBJECT - Anything real or abstract, about which you store both data and operations that manipulate data.

CLASS - An implementation that can be used to create multiple objects with the same attributes and behaviors.

ATTRIBUTES - Identifying characteristics of objects.

METHOD - An activity that reads or manipulates the data of an object.

MESSAGE - Must be sent in order for an object to do something.

ENCAPSULATION - The process of hiding the implementation details of an object from its user.

INHERITANCE - When a subclass automatically receives all of the data and methods of its superclass.

POLYMORPHISM - Allows an instruction to be given to various objects and receive a predictable result.

SUBCLASS - A lower level of a class

SUPERCLASS - A higher level of a class

ACTOR - A role that a user plays with respect to the system

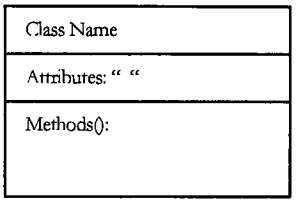
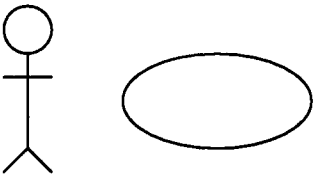
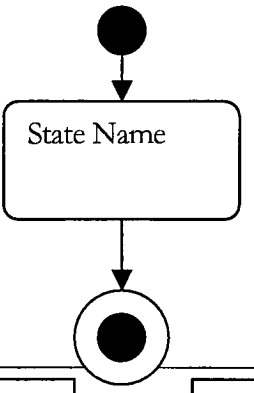
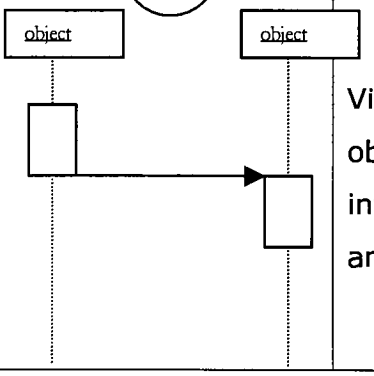
EXTENDING - Describes formally adding steps to a new use case to be used by many

INCLUDING - Describes combining repetitive steps into a use case that can be used by many.

SCENARIO - A sequence of steps describing an interaction between a user and a system

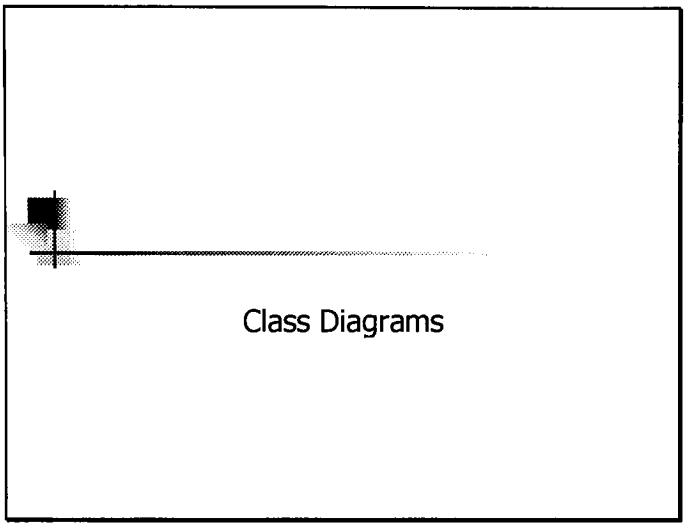
USE CASE - A set of scenarios tied together by a common user goal

APPENDIX C. COURSE SAMPLE DIAGRAMS

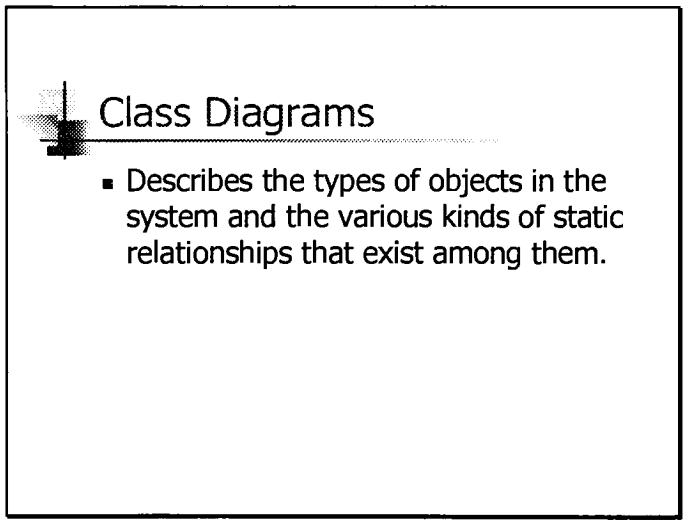
Diagram Name	Basic Diagram	Description
Class Diagram		Shows the entities in a system and how those entities relate to one another.
Use Case Diagram		Shows system usage. A set of a sequence of actions.
State Diagram		Captures the state of an object during a specific time period.
Sequence Diagram		Visualizes how the objects in a system interact with one another over time.

APPENDIX D. COURSE SAMPLE SLIDES

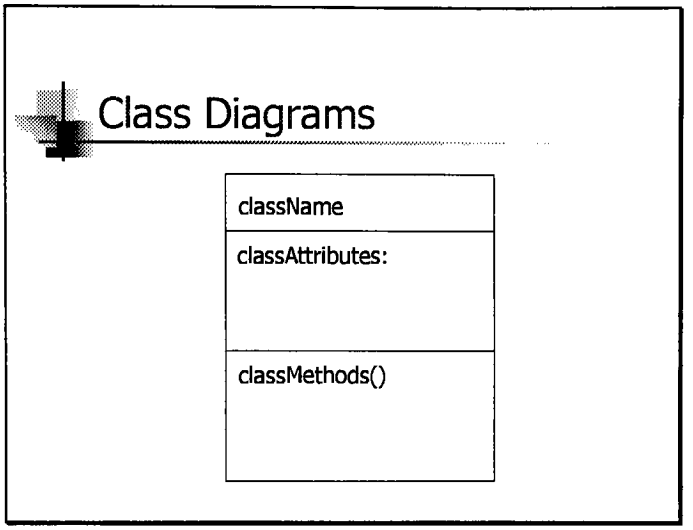
Slide 1



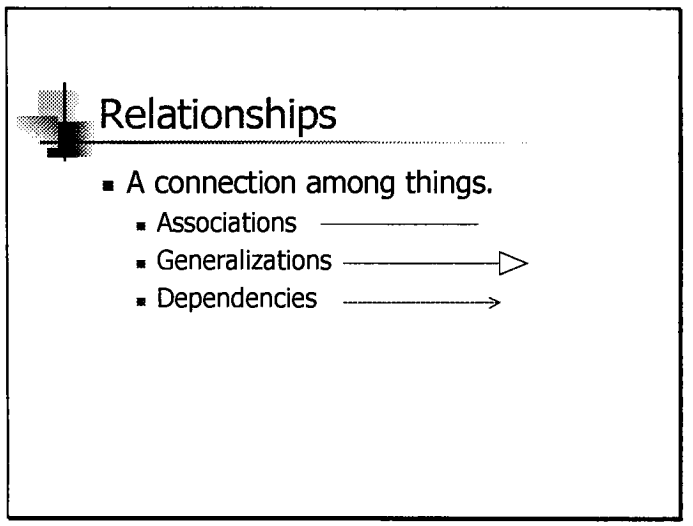
Slide 2



Slide 3



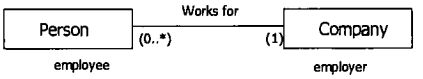
Slide 4



Slide 5

Associations

- Objects of one thing are connected to objects of another thing.
  - Name – describes the nature of the relationship
  - Role – the face that the object plays
  - Multiplicity – how many objects can be connected across an instance (1) (0..1) (0..*) (*) (1..*) (0..4)



Slide 6

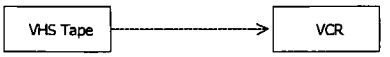
Generalizations

- Relationship between a general thing (superclass) and a more specific thing (subclass)

Slide 7

Dependencies

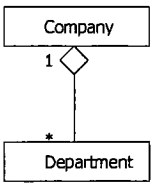
- A using relationship that states that a change in a specification of one thing may affect another thing that uses it, but not necessarily the reverse.



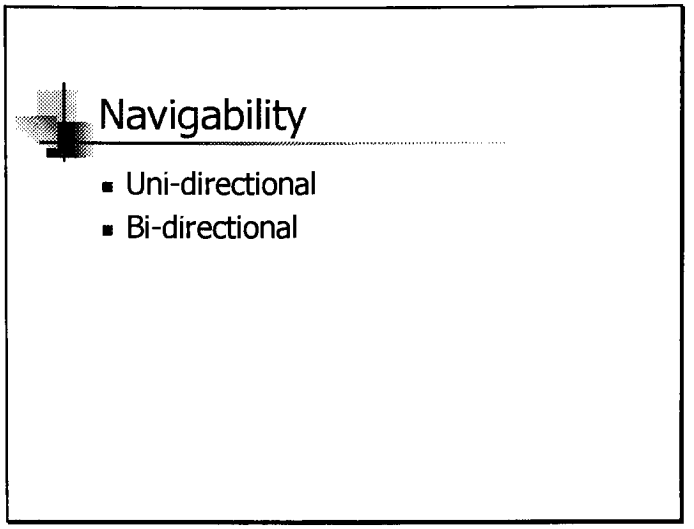
Slide 8

Aggregation

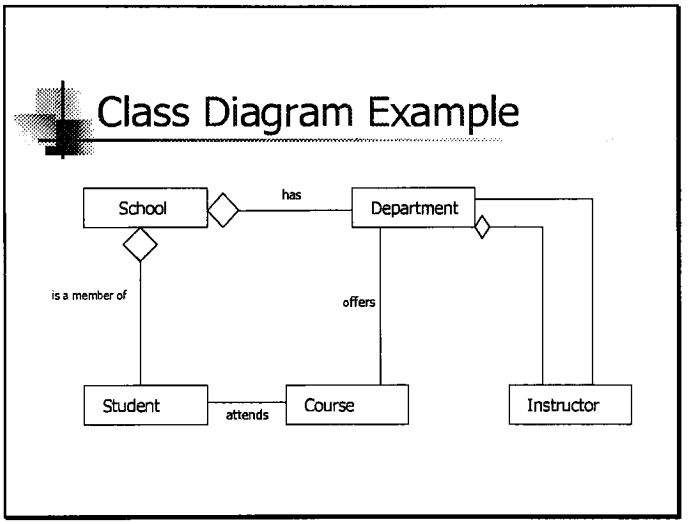
- Whole/part



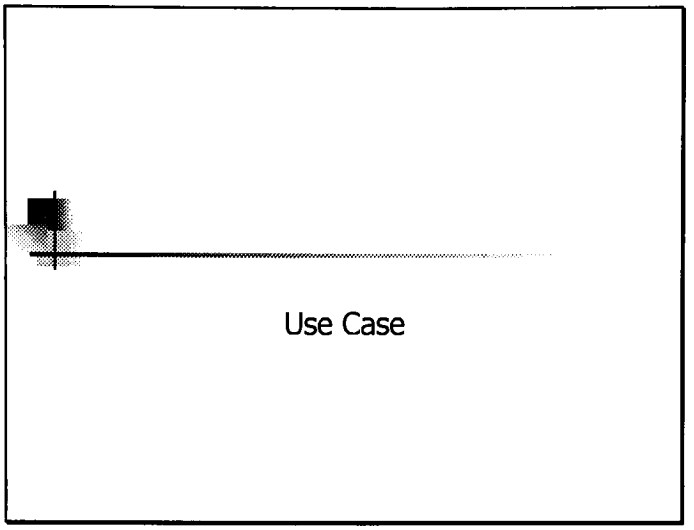
Slide 9



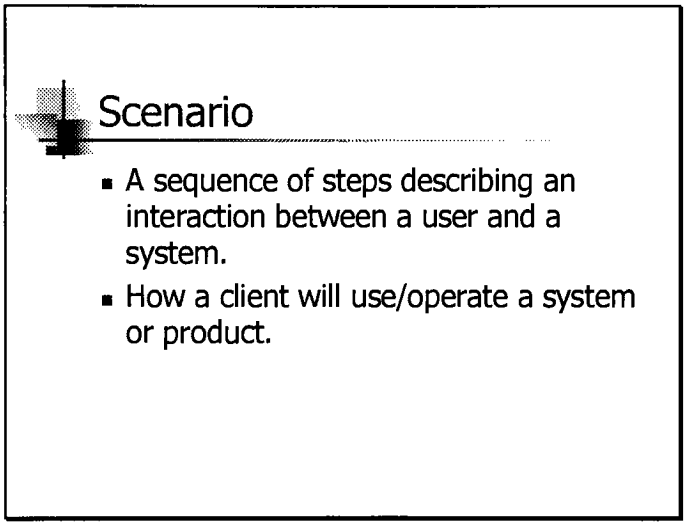
Slide 10



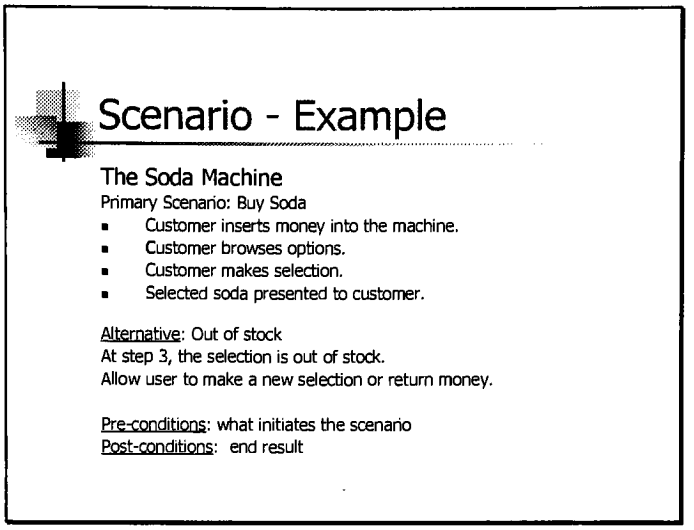
Slide 1



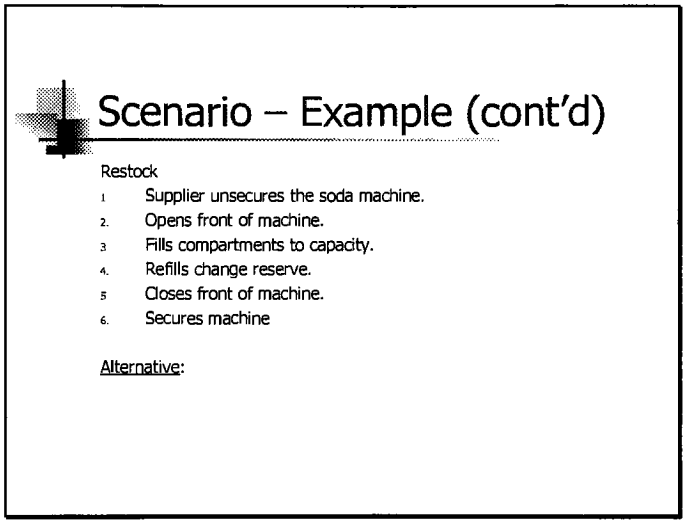
Slide 2



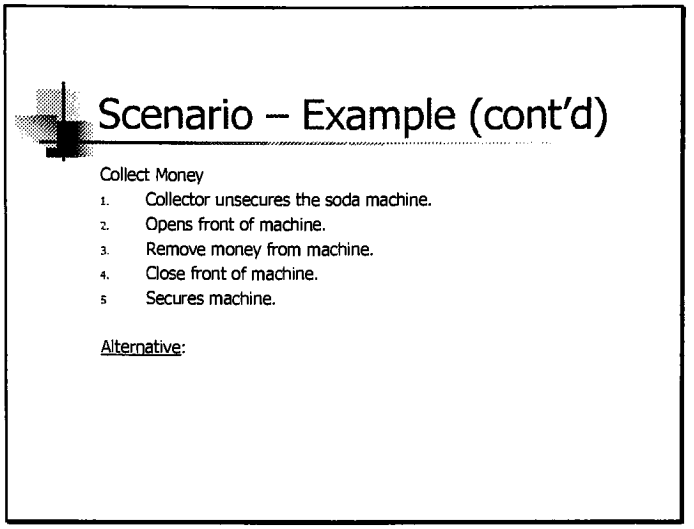
Slide 3



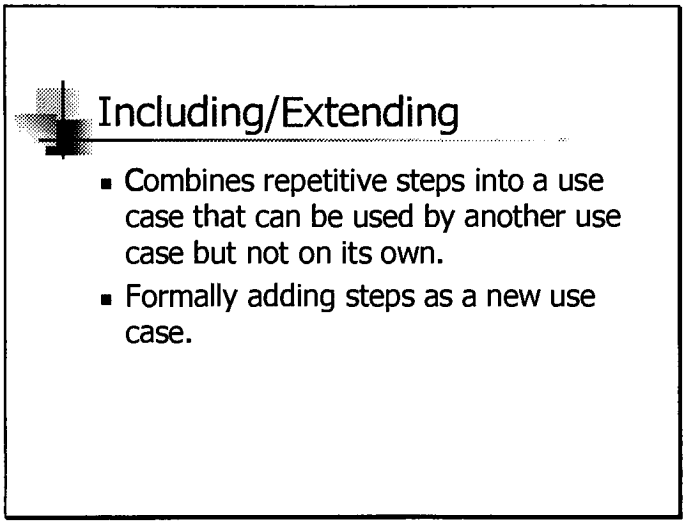
Slide 4



Slide 5



Slide 6



Slide 7

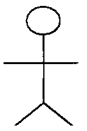
Use Case

- A set of scenarios tied together by a common user goal.
- Information is gained through client interviews and talking with clients and end-users

Slide 8

Use Case Diagrams

- Actor
  - A role that a user plays with respect to the system. Not always a person (piece of hardware, passage of time, another system.)



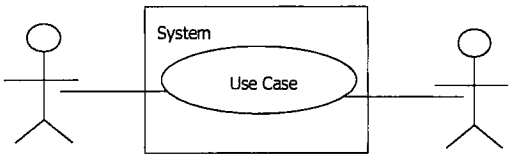
Slide 9

Use Case Diagrams

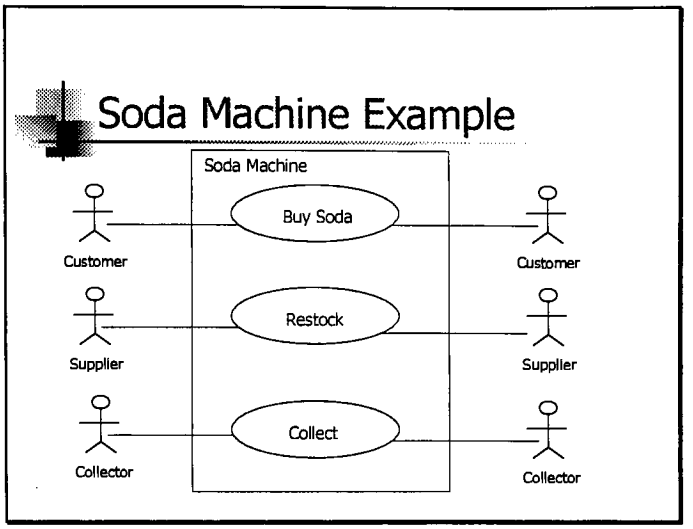
- Associations
  - Shows a relationship between parts of a system.

Slide 10

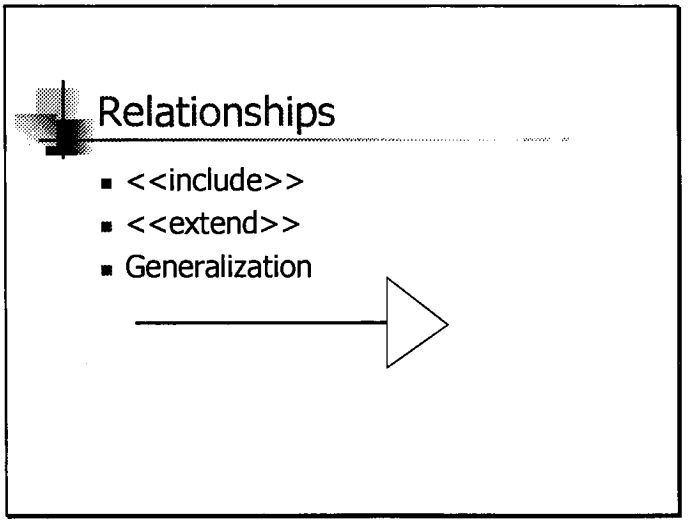
Use Case Diagrams



Slide 11



Slide 12



APPENDIX E. COURSE SAMPLE RATIONAL ROSE LABS

Sample 1
Use Cases

Professor will prepare 3 roles to play for a class interview. The interview will occur with the class as a whole, however the lab exercises will be individual.

An example setting or system for the lab could be a hospital. Roles that the professor would play could be admittance nurse, doctor, and patient.

The professor would start the exercise by giving a brief explanation of what each role does within the system, explaining briefly the process of their currently job. The students would then have an opportunity to ask further detailed questions to the professor. The professor would answer the questions based upon the role they are inquiring about.

Students would then individually create use cases for each of the 3 scenarios discussed with Rational Rose detailing the users role in the system. Students will want to complete the exercises in Chapter 3 of Visual Modeling with Rational Rose 2000 and UML.

Sample 2
Class Diagrams

Based upon the same storyline in Sample 1, the professor will present descriptions of the hospital system. Students will have the ability to ask questions general or specific to the system. The interview will be held with the class as a whole, however the lab exercises will be completed individually.

Students would then individually create classes along with the corresponding attributes, methods and relationships within the classes. Students will want to complete the exercises in Chapter 4 of Visual Modeling with Rational Rose 2000 and UML.

Sample 3
System Architecture

Using a similar storyline as the previous samples, the students will create different views of the system. Logical View, Process View, Use Case View, Implementation View, and Deployment View would be included in the lab.

Students will want to complete the exercises in Chapter 11 of Visual Modeling with Rational Rose 2000 and UML.

APPENDIX F. COURSE SAMPLE QUIZ QUESTIONS

Define the following (3 pts each):

1. Class -

2. Object -

3. Attribute -

4. Method -

5. Encapsulation -

6. Inheritance -

7. Polymorphism -

True/False (2 pts each)

8. T F Objects can never send messages to themselves.

9. T F In a dependent relationship, the dependency is always bi-directional.

10. T F The length of an activation in a sequence diagram is a direct relationship to its duration.

Listings

11. List the four (4) "kinds of things" in the UML": (4 pts)

12. List two (2) benefits of Object-Oriented Programming: (2 pts)

Matching (2 pts each)

13._____	Asynchronous message	a.	Transfers control from one object to another.
14._____	Synchronous message	b.	#
15._____	Simple message	c.	-
16._____	Public	d.	Waits for an answer to that message before it continues.
17._____	Private	e.	+

18._____Protected

f. Does not wait for an answer before continuing.

Diagram (20 pts)

19. Create a class diagram of the below classes. Each class should have two (2) attributes and two (2) methods. All associations should be labeled with a name and should show multiplicity. Assume the maximum capacity for the car is 5.

Classes:
Car
Tires
Passengers

APPENDIX G. COURSE SAMPLE HOMEWORK ASSIGNMENT

Below is an interview between a coach and an analyst/programmer. Pick out at least 2 classes within the interview and list their attributes and methods.

Analyst/Programmer: "Coach, what is basketball all about?"

Coach: " The goal of the game is to shoot the ball through the basket and score more point than your opponent. Each team consists of 5 players: two guards, two forwards, and a center. Each team advances the ball toward the basket w/ the objective of shooting the ball through the basket."

Analyst/Programmer: "How does one advance the ball?"

Coach: "By dribbling and passing."

...

APPENDIX H. COURSE SAMPLE CLASS EXERCISE

Classes to use:

- Employee
- AssistantServer
- AssistantChef
- Bartender
- Server
- Chef
- Manager
- CoatCheckClerk

Relationships to diagram:

- Server to Chef
- Server to Bartender
- Chef to AssistantChef
- Others if times permits

Attributes to distribute:

- name
- address
- ssn
- yrsExp
- hireDate
- salary

Methods to distribute:

- carry()
- pour()
- prepare()
- monitor()
- takeDrinkOrder()
- Others you determine (2 per class)

Bibliography

[i] Fryer, Bronwyn. CNN. "College Computer Science Enrollment Skyrockets". http://www.cnn.com/TECH/computing/9810/22/csboom.idg/. October 1998.

[ii] Melewski, Deborah. "UML Spreads Its Wings". ADTMag.com. September 2000. htt://www.adtmag.com/Pub/article.asp?ArticleID=2656

[iii] OMG.org. "OMG Background Information". http://www.omg.org/news/about/index.htm

[iv] Booch, Jacobson, Rumbaugh. "The Unified Modeling Language User Guide". Addision Wesley Longman, Inc. 1999.

[v] Grant, Delvin A. and Ngwenyama, Ojelanki K. Enterprise modeling for CIM Information Systems Architectures: An Object-Oriented Approach. Computers ind. Engng. Vol. 26 No. 2 pp. 279-293.

[vi] Booch, Jacobson, Rambaugh,. "UML Distilled". Addison Wesley Longman, Inc. 2000.

[vii] Booch, Jacobson, Rambaugh. "The Unified Modeling Language User Guide". Addision Wesley Longman, Inc. 1999. Pg. 3 – 7.

[viii] Booch, Jacobson, Rambaugh. "The Unified Modeling Language User Guide". Addision Wesley Longman, Inc. 1999. Pg. 3 – 7.

[ix] Gray, Clifford F. and Larson, Erik W. Project Management. Irwin McGraw-Hill. 2000.

[x] Walesh, Stuart. "It's Project Management, Stupid!". Journal of Management in Engineering. January/February 1996.

[xi] Walesh, Stuart. "It's Project Management, Stupid!". Journal of Management in Engineering. January/February 1996.

[xii] Griss, Martin L and Jacobson, Ivar. "Approaching the promised land of component reuse". Application Development Trends Magazine. June 2001. http://www.adtmag.com/Pub/article.asp?ArticleID=3680

[xiii] Cantor, Murray. "UML and Communication through the Life Cycle". Software Development Magazine. April 1999. http://www.sdmagazine.com/documents/s=815/sdm9904uml4/sdm9904uml4.htm.

[xiv] Fowler, Martin with Scott, Kendall. "UML Distilled". Addision Wesley Longman, Inc. 2000.

[xv] Cantor, Murray. "UML and Communication through the Life Cycle". Software Development Magazine. April 1999. http://www.sdmagazine.com/documents/s=815/sdm9904uml4/sdm9904uml4.htm.

[xvi] Shannon, Bill, Yarow, Robin and Johnson, .J. "UML can save your next project". UML-Zone. January 1999. http://www.uml-zone.com/articles/sy0199.sy0199.asp.

[xvii] Fowler, Martin with Scott, Kendall. "UML Distilled". Addision Wesley Longman, Inc. 2000.

[xviii] Gray, Clifford F. and Larson, Erik W. Project Management. Irwin McGraw-Hill. 2000.

[xix] Sommerville, Ian. Software Engineering. Addison-Wesley. 1996. Pg 396.

[xx] Alhir, Sinan Si. UML In A Nutshell. O'Reilly & Associates, Inc. 1998. Pg 39 – 68.

[xxi] Booch, Jacobson, Rambaugh. "The Unified Modeling Language User Guide". Addision Wesley Longman, Inc. 1999. Pg. 3 - 35.

[xxii] Alhir, Sinan Si. UML In A Nutshell. O'Reilly & Associates, Inc. 1998. Pg 3 – 37.

[xxiii] Rational Software. www.rational.com.

[xxiv] Quatrani, Terry. Visual Modeling with Rational Rose 2000 and UML. Addison-Wesley Publishing. 2000.